# The Google File System 读后感

GFS 是 google 的一个带有可拓展的新式分布式文件系统，既然是可拓展，顾名思义可用于大量数据，大型数据或者分布式的一个访问系统。当然，也更高效，不然也不会有它的存在在这里。或许都知道 Google 的分布式文件系统，因为没有任何的东西的完美的，最初的分布式文件系统存在着诸多问题，最初的目的是处理众多大数据。

1.大型文件的应用，以至于其大型数据处理能力较好，但对小型数据或文件的识别和处理则并不出色。

2.早期的系统一旦无法识别或正常使用，服务器等组件的缺失损坏都是部件异常等。

3.早期系统文件采用覆盖式的更新，速度极度缓慢，效率低下，无法大面积覆盖。

4.其次则是大家都在说的效率第，交互缓慢且不稳定。由于时代的变迁，需求越来越广泛，普通的交互文件系统已经不能满足新时代数据化时代的高效处理能力，因此新一代的 GFS 则走向了我们的视线。

首先，GFS 的设计目标是一个面向大规模数据密集型应用的、可伸缩的分布式文件系统，同时也提供灾难冗余的能力，为大量客户机提供了高性能的服务。

基于这个目标，GFS 的设计有一下几个前提：首先，组件失效被认为是常态事件，而不是意外事件。其次，以通常的标准衡量，我们的文件非常巨大。第三，绝大部分文件的修改是采用在文件尾部追加数据，而不是覆盖原有数据的方式。第四，应用程序和文件系统 API 的协同设计提高了整个系统的灵活性。

在以上的四个前提下，来看 GFS 的设计目标，可能更快的了解 GFS 的架构。对于海量数据的存储，GFS 将输入的每一份数据分割成 64M 大小的若干个 chunk，每个 chunk 都至少会被备份三份，以降低数据损失的风险。对于每个 chunk 及其备份的位置、数量，都由 master 进行安排，同时 master 会记录下每个 chunk 的名称以及位置信息，用以之后用户的读取。在用户读取数据时，需要从 master 处得到相关 chunk 的名字以及位置信息，得到这些信息后，客户机才能从 chunk 处读取数据。为了避免 master 制约 GFS 的性能，master 不会存储数据，master 只存储 chunk 的名字以及位置信息，并且，chunk 的名字都是被前缀压缩后再进行存储的。同时，在客户机向 master 获取 chunk 的信息时，master 会把之后 chunk 的信息也一并给客户机，之后，除非客户机丢失了这些信息，否则客户便不用在于 master 进行交流了，这样大大的提高了 master 的工作能力。

对于数据的添加，变更操作会在 Chunk 的所有副本上执行。GFS 使用租约（lease）机制来保持多个副本间变更顺序的一致性。在每一份数据的众多 chunk 之中，master 会选择其中的一个 chunk 与其建立租约，这个有租约的叫做主 Chunk ，主 Chunk 对 Chunk 的所有更改操作进行序列化。所有的副本都遵从这个序列进行修改操作。因此，修改操作全局的顺序首先由 Master 节点选择的租约的顺序决定，然后由租约中主 Chunk 分配的序列号决定。设计租约机制的目的是为了最小化 Master 节点的管理负担。

以上便是 GFS 工作的大概流程。